There are 120 points in total, including the bonus question for undergraduates.

1. (20 points) A clique $C$ in a graph $G = (V, E)$ is a subset of $V$ such that for every pair of vertices $u, v \in C$, $(u, v) \in E$.

Consider the following optimization problem to find a maximum size clique.

**Max Clique**

*Input*: graph $G = (V, E)$;

*Output*: a clique $C \subseteq V$ such that $|C|$ is the maximum.

We formulate a corresponding decision problem **Clique** as follows:

**Clique**

*Input*: graph $G = (V, E)$, and integer $k \geq 0$;

*Output*: YES if and only if there is a clique in $G$ of size $\geq k$.

**Claim**: if **Clique** can be solved in polynomial time, so can **Max Clique**. Prove this claim.

**Answers:**

This exercise is similar to the one in the lecture note 5 for Hamiltonian cycle problem.

The proof idea consists two major steps. Let $A$ be a polynomial time algorithm for decision problem **Clique**.

(1) (5 points) To use $A$ to identify the max size of a clique in the graph $G$. We achieve this by calling $A(G, k)$ with $k = 1, 2, \ldots, n$. Since
a single vertex is a clique of size 1, \( A \) will answer "Yes" before giving the "No" answer on some value of \( k \). Assume \( k_0 \) is the least value of \( k \) for \( A(G, k) = "Yes" \) and \( A(G, k_0 + 1) = "No" \).

(2) (10 points) To identify a clique of size \( k_0 \). We achieve this by repeatedly trying \( A \) on modified graph \( G' = G - \{ e \} \) where an arbitrary non-"considered" edge \( e \) is removed from \( G \). That is, we run \( A \) on the modified graph \( G' \). If \( A(G', k_0) = "Yes" \), it implies that the edge \( e \) is not critical to the clique of size \( k_0 \) so it can be removed from \( G \) permanently. Otherwise, \( e \) needs to be reinstated in \( G \) and be marked with "considered". The process repeats to consider all non-"considered" edges. Eventually, only edges constitutes the clique of size \( k_0 \) remain in the graph.

(5 points) The proof also need to argue both steps (1) and (2) take only polynomial time.
2. **(15 points)** Will the claim in question 1 still hold if the inequality \( \geq k \) is replaced with equality \( = k \)? How about being replaced with \( \leq k \)?

**Answers:**

**(8 points)** Yes, because both steps (1) and (2) in question 1 are still correct when inequality \( \geq k \) is replaced with equality \( = k \).

**(7 points)** No, answer to the question if a graph has a clique of size \( \leq k \) is trivial. This is because a single vertex is a clique of size 1, which is \( \leq k \) for any \( k \geq 1 \). The answer is always "Yes" actually for \( k \geq 1 \).
3. **(15 points)** Prove that problem **Clique** is in the class **NP**.

**Answers:**

The answer has to follow the definition of NP class. That is to show that there is a polynomial time verifier $A$ such that for every input $x = (G, k)$, where $G = (V, E)$,

$$G \text{ has a clique of size } \geq k \iff \exists y A(x, y) = 1$$

where $y$ is a certificate.

The answer needs to design a format/content of the certificate (**6 points**) and to show how $A$ verifies the certificate in polynomial time (**9 points**).

One simple design of the certificate is a list of $k$ vertices $y = \{v_1, \ldots, v_k\}$. The verifier $A$ checks the following:

1. $\forall v_i [v_i \in y \Rightarrow v_i \in V]$;
2. $\forall v_i v_j [i \neq j \Rightarrow v_i \neq v_j]$;
3. $\forall v_i v_j [v_i \in y \land v_j \in y \land v_i \neq v_j \Rightarrow (v_i, v_j) \in E]$.

The student then needs to argue that steps (1) through (3) can be done in polynomial steps.
4. **(10 points)** Give a simple reason to explain that if a reduction function $f$ holds for $L_1 \leq L_2$, $f^{-1}$ usually does not hold for $L_2 \leq L_1$, where $f^{-1}$ is the inverse function of $f$.

**Answers:**

There are a couple reasons about function $f$ that may not qualify it to be a mapping for $L_2 \leq L_1$. An answer will need to account for either of them:

1. $f$ may be a many:1 reduction. That is, for two instances $x_1, x_2$, $f(x_1) = f(x_2)$. So the inverse $f^{-1}$ does not exist.

2. Even if $f$ is a 1:1 mapping but not on-to, i.e., there may be $y$ for which there is no such $x$ that $f(x) = y$. Then $f^{-1}(y)$ is not defined.
5. (20 points) Prove that Clique $\leq_p$ Vertex Cover, i.e., problem Clique can be reduced to problem Vertex Cover in polynomial time. (Hint: you may use known facts of reductions).

**Answers:**

This question is to ask the student to use known reductions and the fact that polynomial time reductions compose (i.e., are transitive).

It suffices to say the known facts:

(1) **Clique $\leq_p$ Independent Set**; the reduction $f$ transforms a graph $G = (V, E)$ into the complementary graph $\overline{G} = (V, \overline{E})$ of the same vertex set $V$ but with complementary edges, where $\overline{E} = V \times V - E$, such that $G$ has a clique of size $k$ if and only if $\overline{G}$ has an independent set of size $k$.

(2) **Independent Set $\leq_p$ Vertex Cover**; the reduction $g$ transforms a graph $G = (V, E)$ into the same graph, but the parameter $k$ into $|V| - k$, for the reason that $G$ has an independent set if and only if $G$ has a vertex cover of size $|V| - k$. (Note: here $G$ is different from $G$ in (1).)

(3) polynomial time reductions $\leq_p$ compose; $f$ and $g$ are polynomial time computable. So is the composite transformation $(g \circ f)$ for a reduction for Clique $\leq_p$ Vertex Cover, such that $G$ has a clique of size $k$ if and only if complementary $\overline{G}$ has a vertex cover of size $|V| - k$. 


6. (**20 points, Graduate students only, bonus for undergraduates**) Prove that if $L_1 \leq_p L_2$ and $L_2$ admits an algorithm that runs in time $O(n^{O(\log n)})$, then $L_1$ can be solved in time $O(n^{O(\log n)})$ as well.

**Answers:**

This follows one of the theorems introduced in the lecture. The major task remains the same, i.e., to show the polynomial length of $f(x)$, where $f$ is assumed to be the mapping for the reduction $L_1 \leq_p L_2$.

Assume algorithm $B$ decides $L_2$ and runs in time $O(|y|^c \log |y|)$ for some constant $c \geq 0$, on input $y$ of length $|y|$.

Construct another algorithm $A$ to decide $L_1$ which behaves as follows:

1. $A$ takes the input $x$ and transforms into $f(x)$ using the reduction algorithm $F$ for $L_1 \leq_p L_2$. It takes $O(n^d)$ steps, for some constant $d \geq 0$, where $n = |x|$;

2. $A$ calls $B$ on instance $y$, where $y = f(x)$, $A$ accepts $x$ if and only $B$ accepts $y$. Based on the reduction property, $A$ decides language $L_1$ correctly. In addition, $A$ takes time

$$O(|y|^c \log |y|) = O(|f(x)|^c \log |f(x)|) \leq O(n^{dc \log n^d}) = O(n^{d^2c \log n}) = O(n^{O(\log n)})$$

Summing the times for steps (1) and (2), we still have

$$O(n^d) + O(n^{O(\log n)}) = O(n^{O(\log n)})$$
7. (20 points) Assume two languages $L$ and $L'$ can be polynomially reduced to each other, i.e., $L \leq_p L'$ and $L' \leq_p L$. Prove that, if $L$ is NP-complete, so is $L'$.

**Answers:**

The proof needs to show two conclusions, given the assumption that $L$ is NP-complete:

1. (6 points) To prove that $L'$ is NP-hard;

   Because $L$ is NP-complete, it is NP-hard. That is $\forall J \in NP, J \leq_p L$.
   Because $L \leq_p L'$, by composition of polynomial time reductions, $J \leq_p L'$. So $L'$ is NP-hard.

2. (9 points) To prove that $L'$ is in NP.

   Because $L$ is NP-complete, it is in NP. That is, there is a polynomial time verifier $A$ such that for every $x \in \Sigma^*$,
   
   $$ x \in L \iff \exists y A(x, y) = 1 $$

   Because $L' \leq_p L$ (assuming through mapping function $f$), we have for every $z \in \Sigma^*$,
   
   $$ z \in L' \iff f(z) \in L $$

   Let $x = f(z)$ and combine the above two equivalences, we have
   
   $$ z \in L' \iff \exists y A(f(z), y) = 1 $$

   which can rewritten as
   
   $$ z \in L' \iff \exists y B(z, y) = 1 $$

   where $B$ is an algorithm that first calls the reduction to produce $f(z)$ and then does the work of $A$ on input $f(z)$ and $y$.

   Apparently $B$ runs a polynomial time and it is a checker for language $L'$. So $L'$ is in NP.

   This proves that $L'$ is NP-complete, assuming $L$ is NP-complete.